



Jaguar Compiler Introduction

W. Michael Brown

March 09, 2011



OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY



OAK RIDGE LEADERSHIP COMPUTING FACILITY

Compilers Available on Jaguar

- C, C++, and Fortran compilers available from:
 - Portland Group (PGI)
 - Cray
 - GNU
 - Intel
 - Pathscale
- The choice of compiler will often depend on the code
 - Many codes/compilers do not conform to ISO standards
 - Optimization will be different for different compilers
 - Compiler specific directives/pragmas



Compiling for Compute Nodes

- Calling the compilers directly will compile *serial* code for the *login/service* nodes
- Highly recommended to use the Cray wrappers
 - This will compile code - serial, MPI, OpenMP – for running jobs on the compute nodes
 - Wrappers automatically include paths and libraries for loaded modules (mpi, libsci, etc.)
- The wrappers use the same commands for all vendors – select specific vendors using modules

Language	Wrapper
C	cc
C++	CC
Fortran 77, 90, and 95	ftn

Cray Programming Environment Modules

Vendor	Module
Portland Group (PGI)	PrgEnv-pgi
Cray	PrgEnv-cray
GNU	PrgEnv-gnu
Intel	PrgEnv-intel
Pathscale	PrgEnv-pathscales

To View Available Programming Environments

```
> module avail PrgEnv
```

```
----- /opt/modulefiles -----  
PrgEnv-cray/1.0.2                PrgEnv-intel/3.1.49A(default)  
PrgEnv-cray/3.1.49A(default)    PrgEnv-pathscale/3.1.49A(default)  
PrgEnv-gnu/3.1.49A(default)     PrgEnv-pgi/3.1.49A(default)  
PrgEnv-intel/1.0.0
```

```
> module show PrgEnv-pgi
```

```
-----  
/opt/modulefiles/PrgEnv-pgi/3.1.49A:  
...  
setenv          PE_ENV PGI  
setenv          XTOS_VERSION 3.1.49A  
setenv          XTPE_COMPILE_TARGET linux  
prepend-path   PE_PRODUCT_LIST PGI  
module         load pgi  
module         load xt-libsci  
module         load xt-mpt  
module         load pmi  
module         load xt-asyncpe  
setenv         CRAY_PRGENVPGI loaded
```

Default Programming Environment

- The default environment is for Portland Group (PGI) compilers
- To see currently loaded modules (including those loaded by default at login time) use *module list*

```
> module list
Currently Loaded Modulefiles:
 17) pgi/10.3.0
 18) xt-libsci/10.4.4
 19) pmi/1.0-1.0000.7628.10.2.ss
 20) xt-mpt/4.0.0
 21) xt-pe/2.2.73
 22) xt-asyncpe/3.7
 23) PrgEnv-pgi/2.2.73
```

To Change Programming Environments

- Use `module swap` (or `module unload`; `module load`)

```
> module swap PrgEnv-pgi PrgEnv-gnu
```

Compiler Documentation/ Versions

- The Cray programming environment modules load vendor specific modules from the table below
- The man pages for the wrappers (cc, CC, ftn) do not include vendor specific information
- Instead, use man with the compiler commands directly
 - In most cases, this requires loading the module for the desired environment
- To change the version of a compiler used by the cray wrappers, use module swap, unload, or load with the direct vendor modules
 - `module swap gcc/4.5.1 gcc/4.4.4`

Vendor	Module	C	C++	Fortran
Portland Group	pgi	pgcc	pgCC	pgf77, pgf90, pgf95
Cray	cce	craycc	crayCC	crayftn
GNU	gcc	gcc	g++	gfortran
Intel	intel	icc	icpc	ifort
Pathscale	pathscale	pathcc	pathCC	path90/pathf95

MPI

- The wrappers will use Cray's MPI library (MPT)
- Based on mpich
- Typically, do not need to add compiler include or link flags for MPI when using the Cray wrappers
- SMP aware
 - Shared memory for on-node communications to reduce network traffic

MPI

- The following MPI-2.2 features are currently not supported in MPT:
 - MPI_LONG_DOUBLE data type is not supported
 - canceling of MPI send requests
 - dynamic process management
 - "external32" data representation
 - Process creation commands: MPI_CLOSE_PORT, MPI_OPEN_PORT, MPI_COMM_ACCEPT, MPI_COMM_CONNECT, MPI_COMM_DISCONNECT, MPI_COMM_SPAWN, MPI_COMM_SPAWN_MULTIPLE, MPI_COMM_GET_ATTR - with attribute MPI_UNIVERSE_SIZE, MPI_COMM_GET_PARENT, MPI_LOOKUP_NAME, MPI_PUBLISH_NAME and MPI_UNPUBLISH_NAME

MPI Environment Variables

- Environment variables for tuning MPI performance and debugging can be set
 - MPICH_ENV_DISPLAY
 - If set, then rank 0 will display all MPICH environment variables and their current settings at MPI initialization time. The default is not enabled.
 - MPICH_RANK_REORDER_METHOD
 - Change rank placement for the job
 - Some environment variables are set automatically based on the job size
 - MPICH_MAX_SHORT_MSG_SIZE, MPICH_PTL_OTHER_EVENTS, MPICH_PTL_UNEX_EVENTS, MPICH_UNEX_BUFFER_SIZE
 - MPI I/O Variables

```
> man mpi
```

Dynamic/Shared Libraries

- Typically want to use static links
- If your code does use dynamic/shared libraries:
 - Libraries loaded at runtime must be on the Lustre filesystem
 - Unload the MPT module and load either MPICH2 or SHMEM modules

```
> module unload xt-mpt  
> module load xt-mpich2
```

```
> module unload xt-mpt  
> module load xt-shmem
```

OpenMP

- OpenMP is enabled by *default* with the Cray compilers.
 - Optimizations controlled by `-Othread#`
 - To shut off use `-Othread0` or `-xomp` or `-hnoomp`
 - Autothreading is not enabled by default
- Standard OpenMP environment variables are used
 - `OMP_SCHEDULE`, `OMP_NUM_THREADS`, `OMP_DYNAMIC`, `OMP_NESTED`
 - `OMP_THREAD_STACK_SIZE` is a Cray-specific, nonstandard variable used to change the size of the thread stack from the default size of 16 MB to the specified size.

Module	Enable OMP
PrgEnv-pgi	<code>-mp=nonuma</code>
PrgEnv-cray	<code>-homp</code>
PrgEnv-gnu	<code>-fopenmp</code>
PrgEnv-intel	<code>-openmp</code>
PrgEnv-pathscales	<code>-mp</code>

General Optimization Flags

- The Cray optimization flags are turned on by default
- Can sometimes benefit from tuning flags, compiler directives, and code for
 - Loop unrolling
 - Function Inlining
 - Vectorization
 - Interprocedural Analysis
 - -Mipa=fast,inline (on pgi)
 - malloc

Module	Flag
PrgEnv-pgi	-fast
PrgEnv-cray	<none>
PrgEnv-gnu	-O3
PrgEnv-intel	-O2 -xW -axP
PrgEnv-pathscale	-Ofast

Compiler Feedback

- Most compilers will give feedback to help in tuning code performance

- Loop Unrolling
- Loop Vectorization
- Inlining
- ...

Module	Flag
PrgEnv-pgi	-Minfo=all -Mneginfo=all
PrgEnv-cray	-hlist=m (C, C++) -rm (<i>Fortran</i>)
PrgEnv-gnu	-ftree-vectorizer-verbose= <i>n</i>
PrgEnv-intel	-vec-report1
PrgEnv-pathscale	-LNO:simd_verbose=ON

Profiling and Instrumenting Code

- CrayPAT
 - Apprentice2 GUI
- VampirTrace
 - Vampir GUI
- Get counts for FLOPS, Cache Hits/Misses, TLB Hits/Misses, etc.
- Get MPI statistics
 - How many times are routines called and how much data
 - Identify Load imbalance
- I/O statistics
 - How much time spent in I/O
 - How many processes
- Math Libraries
- ...
-

CrayPAT Groups

- biolib - Cray Bioinformatics library routines
- blacs - Basic Linear Algebra communication subprograms
- **blas** - Basic Linear Algebra subprograms
- **fftw** - Fast Fourier Transform library (64-bit only)
- hdf5 - manages extremely large and complex data collections
- **heap** - dynamic heap
- **io** - includes stdio and sysiogroups
- **lapack** - Linear Algebra Package
- lustre - LustreFile System
- **math** - ANSI math
- **mpi** - MPI
-
- netcdf - network common data form (manages array-oriented scientific data)
- omp - OpenMP API
- omp-rtl - OpenMP runtime library portals Lightweight message passing API
- pthreads - POSIX threads
- **scalapack** - Scalable LAPACK
- shmem - SHMEM
- stdio - all library functions that accept or return the FILE* construct
- sysio - I/O system calls
- system system calls
-

CrayPAT Simple Example

- Automatic Program Analysis

- 2-stage Process
- First gather sampling for line number profile
 - Light-weight
 - Guides what should and should not be instrumented
- Second gather instrumentation (-g mpi,io,blas,lapack,math...)
 - Hardware counters
 - MPI message passing information
 - I/O information
 - Math Libraries
 - ...

```
> module load xt-craypat
> module load apprentice2
> module load xt-papi
> make clean
> make
> pat_build -O apa my_program
> aprun -n <numproc> my_program+pat
> pat_report -T -o report1.txt
my_program+pat+PID-nodesdt.xf
> pat_build -O my_program+pat+PID-
nodesdt.apa
> aprun -n <numproc> my_program+apa
> pat_report -T -o report2.txt
my_program+apa+PID2-nodesdt.xf
> app2 my_program+apa+PID2-nodesdt.ap2
```

VampirTrace Simple Example

- With VampirTrace, you must switch the Cray compiler wrappers with VampirTrace wrappers (available for all compilers with most versions)

- vtcc
- vtc++
- vtf77
- vtf90

- Additionally, the **vt** flag must be specified for the wrappers

- **-vt:seq** for sequential applications
- **-vt:mpi** for applications using mpi
- **-vt:mt** for threaded applications (pthreads, OpenMP)
- **-vt:hyb** for multithreaded mpi applications

```
> module load vampirtrace
> make clean
> make -f Makefile.vt
> aprun -n <numproc> my_program
```

Debugging

- Compile with `-g`
 - Often helpful to turn off optimization with `-O0`
- Many debugging checks can be turned on with additional compiler flags and/or environment variables
 - For PGI:
 - `-gopt` generates debug information in the presence of optimization
 - `-Mbounds` adds array bounds checking
 - `-Ktrap=fp` Trap floating point exceptions
 - `-Mchkptr` Checks for unintended dereferencing of null pointers
 - For MPI:
 - `MPICH_MEMCPY_MEM_CHECK` – check memcpy source/destination overlap
- Parallel Debuggers Available:

```
> module load ddt
> ddt
```

```
> module load totalview
> totalview ./a.out core
```

```
> module load totalview
> totalview aprun -a -n256 ./a.out
```

Resources for Users: More Information

This workshop

- Talks tomorrow on Cray MPT, optimization for the XT5, Cray Compilers, PGI Compilers, CrayPAT, DDT, TotalView

Training Material and Workshop Archives

- Training and talks from previous workshops available on the OLCF web pages (<http://www.olcf.ornl.gov/>)
 - Keep in mind – hardware and software changing fast

Contact us

- help@olcf.ornl.gov

•

•

Resources for Users: More Information

Software Optimization Guide for AMD64 Processors

http://support.amd.com/us/Processor_TechDocs/25112.PDF

PGI Compilers for XT5

<http://www.pgroup.com/resources/docs.htm>

Cray Compilers

<http://docs.cray.com/>

Gnu Compilers

<http://gcc.gnu.org/onlinedocs/>

Intel Compilers

<http://software.intel.com/en-us/intel-compilers/>

Pathscale Compilers

http://www.pathscale.com/Cray_Documentation/

